AMENDMENTS TO THE CLAIMS

Claims 1-5, 7-19 and 21-66 were pending at the time of the Action.

Claims 11-14, 25-28 and 44-47 are canceled in this Response.

Claims 1, 3-5, 7-9, 15, 17-19, 21-23, 29-31, 34-36, 39-41, 48-50, 53-55, 58-60, 63 and 65 are amended in this Response.

Claims 1, 7-9, 15, 21-23, 63 and 65 are independent claims.

Accordingly, claims 1-5, 7-10, 15-19, 21-24, 29-43 and 48-66 remain pending.

Claim 1 (currently amended): A method for creating a binary tree data structure, the data structure embodied in a computer-readable medium, from an ordered list of at least four elements, each element having an associated value in the list, comprising:

determining whether the list has an even or odd number of elements;

separating the list into left side groupings and right side groupings based on whether the list has an even or odd number of elements, the groupings being separated by a parent node defined by a median of the list, wherein the median is a left element of two middle values of the list when the list has an even number of elements, or the median is a middle value element of the list when the list has an odd ~~does not have an even~~ number of elements;

~~inserting~~ creating left side descendent nodes ~~into~~ of the binary tree by successively finding a median of each left side grouping and linking each found median to the previous median;

~~inserting~~ creating right side descendent nodes ~~into~~ of the binary tree by successively finding a median of each right side grouping and linking each found median to the previous median;

wherein when a grouping has an even number of elements, the median is a left element of two middle values of the grouping;

wherein when a grouping ~~does not have an even~~ has an odd number of number of elements, the median is a middle value element of the grouping; and

wherein the elements of the lists include logged events.

Claim 2 (original): A computer-readable medium having stored thereon computer-executable .instructions for performing the method of claim 1.

Claim 3 (currently amended): The method of claim 1, wherein each element in the list includes a pointer to a corresponding node of a plurality of nodes in a partially assembled binary tree, wherein each node has a left child pointer, and wherein ~~inserting~~ creating the left side nodes farther comprises assigning a value to the left child pointer of at least one of the nodes.

Claim 4 (currently amended): The method of claim 1; wherein each element in the list includes a pointer to a corresponding node of a plurality of nodes in a partially assembled binary tree, wherein each node has a right child pointer, and wherein ~~inserting~~ creating the right side nodes further comprises assigning a value to the right child pointer of at least one of the nodes.

Claim 5 (currently amended): The method of claim 1, wherein ~~inserting~~ creating the left side descendent nodes comprises inserting the left side descendent nodes into a partially assembled version of the binary tree, wherein ~~inserting~~ creating the right side descendent nodes comprises ~~inserting~~ creating the right side descendent nodes into the partially assembled version of the binary tree, and wherein the list is a linked list that acts as a wrapper around the partially assembled version of the binary tree.

Claim 6 (canceled).

Claim 7 (currently amended): A method for creating a binary tree data structure, the data structure embodied in a computer-readable medium, from an ordered list of at least four elements, each element having an associated value in the list, comprising:

determining whether the list has an even or odd number of elements;

separating the list into left side groupings and right side groupings based on whether the list has an even or odd number of elements, the groupings being groupings separated by a parent node defined by a median of the list, wherein the median is a left element of two middle values of the list when the list has an even number of elements, or the median is a middle value element of the list when list has an odd ~~does not have an even~~ number of elements;

~~inserting~~ creating left side descendent nodes ~~into~~ of the binary tree by successively finding a median of each left side grouping and linking each found median to the previous median;

~~inserting~~ creating right side descendent nodes ~~into~~ of the binary tree by successively finding a median of each right side grouping and linking each found median to the previous median;

wherein when a grouping has an even number of elements, the median is a left element of two middle values of the grouping;

wherein when a grouping ~~does not have an even~~ has an odd number of number of elements, the median is a middle value element of the grouping; and

wherein the elements of the list include data representing number of times one or more threads of execution have passed through one or more code modules.

Claim 8 (currently amended): A method for creating a binary tree data structure, the data structure embodied in a computer-readable medium, from an ordered list of at least four elements, each element having an associated value in the list, comprising:

determining whether the list has an even or odd number of elements;

separating the list into left side groupings and right side groupings based on whether the list has an even or odd number of elements, the groupings being separated by a parent node defined by a median of the list, wherein the median is a left element of two middle values of the list when the list has an even number of elements, or the median is a middle value element of the list when the list has an odd ~~does not have an even~~ number of elements;

~~inserting~~ creating left side descendent nodes ~~into~~ of the binary tree by successively finding a median of each left side grouping and linking each found median to the previous median;

~~inserting~~ creating right side descendent nodes ~~into~~ of the binary tree by successively finding a median of each right side grouping and linking each found median to the previous median;

wherein when a grouping has an even number of elements, the median is a left element of two middle values of the grouping;

wherein when a grouping ~~does not have an even~~ has an odd number of number of elements, the median is a middle value element of the grouping; and

wherein the ~~inserted~~ created right and left descendant nodes include data representing a number of times one or more threads of execution have passed through one or more code modules.

Claim 9 (currently amended): A method for creating a binary tree data structure, the data structure embodied in a computer-readable medium, from an ordered list of at least four elements, each element having an associated value in the list, comprising:

separating the list into left side groupings and right side groupings based on whether the list has an even or odd number of elements, the groupings being separated by a parent node defined by a median of the list, wherein the median is a left element of two middle values of the list when the list has an even number of elements, or the median is a middle value element of the list when the list has an

odd ~~does not have an even~~ number of elements;

~~inserting~~ creating left side descendent nodes ~~into~~ of the binary tree by successively finding a median of each left side grouping and linking each found median to the previous median;

~~inserting~~ creating right side descendent nodes ~~into~~ of the binary tree by successively finding a median of each right side grouping and linking each found median to the previous median;

wherein when a grouping has an even number of elements, the median is a left element of two middle values of the grouping;

wherein when a grouping ~~does not have an even~~ has an odd number of number of elements, the median is a middle value element of the grouping; and

wherein the ~~inserted~~ created right and left descendant nodes include one or more pointers to data representing a number of times one or more threads of execution have passed through one or more code modules.

Claim 10 (original): The method of claim 1, wherein the list is an ordered linked list.

Claim 11 (canceled)

Claim 12 (canceled)

Claim 13 (canceled)

Claim 14 (canceled)

Claim 15 (currently amended): A method for creating a binary tree data structure, the data structure embodied in a computer-readable medium, from an ordered list of at least four elements, each element having an associated value in

the list, comprising:

determining whether the list has an even or odd number of elements;

separating the list into left side groupings and right side groupings based on whether the list has an even or odd number of elements, the groupings being separated by a parent node defined by a median of the list, wherein the median is a left element of two middle values of the list when the list has an even number of elements, or the median is a middle value element of the list when the list has an odd ~~does not have an even~~ number of elements;

~~inserting~~ creating right side descendent nodes ~~into~~ of the binary tree by successively finding a median of each right side grouping and linking each found median to the previous median;

~~inserting~~ creating left side descendent nodes ~~into~~ of the binary tree by successively finding a median of each left side grouping and linking each found median to the previous median;

wherein when a grouping has an even number of elements, the median is a right element of two middle values of the grouping;

wherein when a grouping ~~does not have an even~~ has an odd number of number of elements, the median is a middle value element of the grouping; and

wherein the elements of the list include logged events.

Claim 16 (original): A computer-readable medium having stored thereon computer-executable instructions for performing the method of claim 15.

Claim 17 (currently amended): The method of claim 15, wherein each element in the list includes a pointer to a corresponding node of a plurality of nodes in a partially assembled binary tree, wherein each node has a right child pointer, and wherein ~~inserting~~ creating the right side nodes further comprises assigning a value to the right child pointer of at least one of the nodes.

Claim 18 (currently amended): The method of claim 15, wherein each element in the list includes a pointer to a corresponding node of a plurality of nodes in a partially assembled binary tree, wherein each node has a left child pointer, and wherein ~~inserting~~ creating the left side nodes further comprises assigning a value to the left child pointer of at least one of the nodes.

Claim 19 (currently amended): The method of claim 15, wherein ~~inserting~~ creating the right side descendent nodes comprises ~~inserting~~ creating the right side descendent nodes into a partially assembled version of the binary tree, wherein ~~inserting~~ creating the left side descendent nodes comprises ~~inserting~~ creating the left side descendent nodes into the partially assembled version of the binary tree, and wherein the list is a linked list that acts as a wrapper around the partially assembled version of the binary tree.

Claim 20 (canceled).

Claim 21 (currently amended): A method for creating a binary tree data structure, the data structure embodied in a computer-readable medium, from an ordered list of at least four elements, each element having an associated value in the list, comprising:

determining whether the list has an even or odd number of elements;

separating the list into left side groupings and right side groupings based on whether the list has an even or odd number of elements, the groupings being separated by a parent node defined by a median of the list, wherein the median is a left element of two middle values of the list when the list has an even number of elements, or the median is a middle value element of the list when the list has an odd ~~does not have an even~~ number of elements;

~~inserting~~ creating right side descendent nodes ~~into~~ of the binary tree by successively finding a median of each right side grouping and linking each found

median to the previous median;

~~inserting~~ creating left side descendent nodes ~~into~~ of the binary tree by successively finding a median of each left side grouping and linking each found median to the previous median;

wherein when a grouping has an even number of elements, the median is a right element of two middle values of the grouping;

wherein when a grouping ~~does not have an even~~ has an odd number of number of elements, the median is a middle value element of the grouping; and

wherein the elements of the list include data representing a number of times one or more threads of execution have passed through one or more code modules.

Claim 22 (currently amended): A method for creating a binary tree data structure, the data structure embodied in a computer-readable medium, from an ordered list of at least four elements, each element having an associated value in the list, comprising:

determining whether the list has an even or odd number of elements;

separating the list into left side groupings and right side groupings based on whether the list has an even or odd number of elements, the groupings being separated by a parent node defined by a median of the list, wherein the median is a left element of two middle values of the list when the list has an even number of elements, or the median is a middle value element of the list when the list has an odd ~~does not have an even~~ number of elements;

~~inserting~~ creating right side descendent nodes ~~into~~ of the binary tree by successively finding a median of each right side grouping and linking each found median to the previous median;

~~inserting~~ creating left side descendent nodes ~~into~~ of the binary tree by successively finding a median of each left side grouping and linking each found median to the previous median;

wherein when a grouping has an even number of elements, the median is a

right element of two middle values of the grouping;

wherein when a grouping ~~does not have an even~~ has an odd number of number of elements, the median is a middle value element of the grouping; and

wherein the ~~inserted~~ created right and left descendant nodes include data representing a number of times one or more threads of execution have passed through one or more code modules.

Claim 23 (currently amended): A method for creating a binary tree data structure, the data structure embodied in a computer-readable medium, from an ordered list of at least four elements, each element having an associated value in the list, comprising:

determining whether the list has an even or odd number of elements;

separating the list into left side groupings and right side groupings based on whether the list has an even or odd number of elements, the groupings being separated by a parent node defined by a median of the list, wherein the median is a left element of two middle values of the list when the list has an even number of elements, or the median is a middle value element of the list when the list has an odd ~~does not have an even~~ number of elements;

~~inserting~~ creating right side descendent nodes ~~into~~ of the binary tree by successively finding a median of each right side grouping and linking each found median to the previous median;

~~inserting~~ creating left side descendent nodes ~~into~~ of the binary tree by successively finding a median of each left side grouping and linking each found median to the previous median;

wherein when a grouping has an even number of elements, the median is a right element of two middle values of the grouping;

wherein when a grouping ~~does not have an even~~ has an odd number of number of elements, the median is a middle value element of the grouping; and

wherein the ~~inserted~~ created right and left descendant nodes include one or

more pointers to data representing a number of times one or more threads of execution have passed through one or more code modules.

Claim 24 (original): The method of claim 15, wherein the list is an ordered linked list.

Claim 25 (canceled)

Claim 26 (canceled)

Claim 27 (canceled)

Claim 28 (canceled).

Claim 29 (currently amended): The method of claim 7, wherein each element in the list includes a pointer to a corresponding node of a plurality of nodes in a partially assembled binary tree, wherein each node has a left child pointer, and wherein ~~inserting~~ creating the left side nodes further comprises assigning a value to the left child pointer of at least one of the nodes.

Claim 30 (currently amended): The method of claim 7 wherein each element in the list includes a pointer to a corresponding node of a plurality of nodes in a partially assembled binary tree, wherein each node has a right child pointer, and wherein ~~inserting~~ creating the right side nodes further comprises assigning a value to the right child pointer of at least one of the nodes.

Claim 31 (currently amended): The method of claim 7, wherein ~~inserting~~ creating the left side descendent nodes comprises ~~inserting~~ creating the left side descendent nodes into a partially assembled version of the binary tree, wherein

~~inserting~~ creating the right side descendent nodes comprises ~~inserting~~ creating the right side descendent nodes into the partially assembled version of the binary tree, and wherein the list is a linked list that acts as a wrapper around the partially assembled version of the binary tree.

Claim 32 (previously presented): The method of claim 7, wherein the list is an ordered linked list.

Claim 33 (previously presented): A computer-readable medium having stored thereon computer-executable instructions for performing the method of claim 7.

Claim 34 (currently amended): The method of claim 8, wherein each element in the list includes a pointer to a corresponding node of a plurality of nodes in a partially assembled binary tree, wherein each node has a left child pointer, and wherein ~~inserting~~ creating the left side nodes further comprises assigning a value to the left child pointer of at least one of the nodes.

Claim 35 (currently amended): The method of claim 8, wherein each element in the list includes a pointer to a corresponding node of a plurality of nodes in a partially assembled binary tree, wherein each node has a right child pointer, and wherein ~~inserting~~ creating the right side nodes further comprises assigning a value to the right child pointer of at least one of the nodes.

Claim 36 (currently amended): The method of claim 8, wherein ~~inserting~~ creating left side descendent nodes comprises inserting the left side descendent nodes into a partially assembled version of the binary tree, wherein ~~inserting~~ creating the right side descendent nodes comprises ~~inserting~~ creating the right side descendent nodes into the partially assembled version of the binary tree, and

wherein the list is a linked list that acts as a wrapper around the partially assembled version of the binary tree.

Claim 37 (previously presented): The method of claim 8, wherein the list is an ordered linked list.

Claim 38 (previously presented): A computer-readable medium having stored thereon computer-executable instructions for performing the method of claim 8.

Claim 39 (currently amended): The method of claim 9, wherein each element in the list includes a pointer to a corresponding node of a plurality of nodes in a partially assembled binary tree, wherein each node has a left child pointer, and wherein ~~inserting~~ creating the left side nodes further comprises assigning a value to the left child pointer of at least one of the nodes.

Claim 40 (currently amended): The method of claim 9, wherein each element in the list includes a pointer to a corresponding node of a plurality of nodes in a partially assembled binary tree, wherein each node has a right child pointer, and wherein ~~inserting~~ creating the right side nodes further comprises assigning a value to the right child pointer of at least one of the nodes.

Claim 41 (currently amended): The method of claim 9, wherein ~~inserting~~ creating the left side descendent nodes comprises ~~inserting~~ creating the left side descendent nodes into a partially assembled version of the binary tree, wherein ~~inserting~~ creating the right side descendent nodes comprises ~~inserting~~ creating the right side descendent nodes into the partially assembled version of the binary tree, mad wherein the list is a linked list that acts as a wrapper around the partially assembled version of the binary tree.

Claim 42 (previously presented): The method of claim 9, wherein the list is an ordered linked list.

Claim 43 (previously presented): A computer-readable medium having stored thereon computer-executable instructions for performing the method of claim 9.

Claim 44 (canceled)

Claim 45 (canceled)

Claim 46 (canceled)

Claim 47 (canceled)

Claim 48 (currently amended): The method of claim 21, wherein each element in the list includes a pointer to a corresponding node of a plurality of nodes in a partially assembled binary tree, wherein each node has a right child pointer, and wherein ~~inserting~~ creating the right side nodes further comprises assigning a value to the right child pointer of at least one of the nodes..

Claim 49 (currently amended): The method of claim 21, wherein each element in the list includes a pointer to a corresponding node of a plurality of nodes in a partially assembled binary tree, wherein each node has a left child pointer, and wherein ~~inserting~~ creating the left side nodes further comprises assigning a value to the left child pointer of at least one of the nodes.

Claim 50 (currently amended): The method of claim 21, wherein ~~inserting~~

creating the right side descendent nodes comprises ~~inserting~~ creating the right side descendent nodes into a partially assembled version of the binary tree, wherein ~~inserting~~ creating the left side descendent nodes comprises ~~inserting~~ creating the left side descendent nodes into the partially assembled version of the binary tree, and wherein the list is a linked list that acts as a wrapper around the partially assembled version of the binary tree.

Claim 51 (previously presented): The method of claim 21, wherein the list is an ordered linked list.

Claim 52 (previously presented): A computer-readable medium having stored thereon computer-executable instructions for performing the method of claim 21.

Claim 53 (currently amended): The method of claim 22, wherein each element in the list includes a pointer to a corresponding node of a plurality of nodes in a partially assembled binary tree, wherein each node has a right child pointer, and wherein ~~inserting~~ creating the right side nodes further comprises assigning a value to the right child pointer of at least one of the nodes.

Claim 54 (currently amended): The method of claim 22, wherein each element in the list includes a pointer to a corresponding node of a plurality of nodes in a partially assembled binary tree, wherein each node has a left child pointer, and wherein ~~inserting~~ creating the left side nodes further comprises assigning a value to the left child pointer of at least one of the nodes.

Claim 55 (currently amended): The method of claim 22, wherein ~~inserting~~ creating the right side descendent nodes comprises ~~inserting~~ creating the right side descendent nodes into a partially assembled version of the binary tree, wherein

~~inserting~~ creating the left side descendent nodes comprises ~~inserting~~ creating the left side descendent nodes into the partially assembled version of the binary tree, and wherein the list is a linked list that acts as a wrapper around the partially assembled version of the binary tree.

Claim 56 (previously presented): The method of claim 22, wherein the list is an ordered linked list.

Claim 57 (previously presented): The computer-readable medium having stored thereon computer-executable instructions for performing the method of claim 22.

Claim 58 (currently amended): The method of claim 23, wherein each element in the list includes a pointer to a corresponding node of a plurality of nodes in a partially assembled binary tree, wherein each node has a right child pointer, and wherein ~~inserting~~ creating the right side nodes further comprises assigning a value to the right child pointer of at least one of the nodes.

Claim 59 (currently amended): The method of claim 23, wherein each element in the list includes a pointer to a corresponding node of a plurality of nodes in a partially assembled binary tree, wherein each node has a left child pointer, and wherein ~~inserting~~ creating the left side nodes further comprises assigning a value to the left child pointer of at least one of the nodes.

Claim 60 (currently amended): The method of claim 23, wherein ~~inserting~~ creating the right side descendent nodes comprises ~~inserting~~ creating the right side descendent nodes into a partially assembled version of the binary tree, wherein ~~inserting~~ creating the left side descendent nodes comprises ~~inserting~~ creating the left side descendent nodes into the partially assembled version of the binary tree,

and wherein the list is a linked list that acts as a wrapper around the partially assembled version of the binary tree.

Claim 61 (previously presented): The method of claim 23, wherein the list is an ordered linked list.

Claim 62 (previously presented): A computer-readable medium having stored thereon computer-executable instructions for performing the method of claim 23.

Claim 63 (currently amended): A method for creating a binary tree data structure, the data structure embodied in a computer-readable medium, from an ordered list of at least four elements, each element having an associated value in the list, comprising:

(a) determining whether the list has an even or odd number of elements;

(b) designating a median element of the list as a parent element based on whether the list has an even or odd number of elements, wherein the parent element divides the list into left side groupings and right side groupings, wherein the median is a right element of two middle values of the list when the list has an even number of elements, or the median is a middle value element of the list when the list has an odd ~~does not have an even~~ number of elements;

(c) successively subdividing the right side groupings of the list and linking each successive median element with a previous median element, thereby creating right side descendent nodes in the binary tree;

(d) once each right side grouping has been exhausted as a result of step (c), stepping back up the tree through each successive ancestor node until reaching an element having left side groupings in the list, and, upon reaching an element having a left side grouping in the list, proceeding to step (e);

(e) subdividing the left side groupings and linking a median element of the

left side grouping with the element reached in step (d), thereby creating a left side descendent of the binary tree;

(f) if the left side descendent of step (e) has a right side grouping in the list, repeating step (c) for the right side grouping;

(g) if the left side descendent of step (e) has no right side groupings, but has a left side grouping, repeating step (e) for the left side grouping;

wherein the median element of a grouping is a right element of two middle values of the grouping when the grouping has an even number of elements, or the median is a middle value element of the grouping when the list ~~does not have an even~~ has an odd number of number of elements; and

wherein the elements of the list include data representing number of times one or more threads of execution have passed through one or more code modules.

Claim 64 (previously presented): A computer-readable medium having stored thereon computer-executable instructions for performing the method of claim 63.

Claim 65 (currently amended): A method for creating a binary tree data structure, the data structure embodied in a computer-readable medium, from an ordered list of at least four elements, each element having an associated value in the list, comprising:

(a) determining whether the list has an even or odd number of elements;

(b) designating a median element of the list as a parent element based on whether the list has an even or odd number of elements, wherein the parent element divides the list into left side groupings and right side groupings, wherein the median is a right element of two middle values of the list when the list has an even number of elements, or the median is a middle value element of the list when the list has an odd ~~does not have an even~~ number of elements;

(c) determining if there are elements to the right of the parent element;

(d) if there are no elements to the right of the parent element, proceeding to step (h);

(e) for the elements that are to the right of the parent element, finding a median element;

(f) linking the median element of step (e) to the parent element so that the median element is a child of the parent element;

(g) repeating steps (d) and (e), wherein the child element of step (f) is now treated as the parent element in steps (d) and (e);

(h) locating a next element up on the tree that has elements to the left of it and treating the element as a parent element in step (i);

(i) finding a median element of the elements to the left of the parent element from step (h);

(j) linking the median element of step (i) to the parent element of step (h), wherein the median element is a child of the parent;

(k) repeating steps (d) through (g), wherein the child element of step (j) is treated as the parent element in step (d);

wherein the median element of a grouping is a right element of two middle values of the grouping when the grouping has an even number of elements, or the median is a middle value element of the grouping when the list ~~does not have an even~~ has an odd number of number of elements; and

wherein the elements of the list include data representing number of times one or more threads of execution have passed through one or more code modules.

Claim 66 (previously presented): A computer-readable medium having stored thereon computer-executable instructions for performing the method of claim 65.